ALGOL型プログラム言語における意味論と証明論 (プログラムの基礎理論)

AUTHOR(S):

五十嵐, 滋

CITATION:

五十嵐, 滋. ALGOL型プログラム言語における意味論と証明論 (プログラムの基礎理論). 数理解析研究所講究録 1971, 119: 81-111

ISSUE DATE:

1971-07

URL:

http://hdl.handle.net/2433/106463

数理解析研究所講究録
119巻 1971年 81-111

ALGOL型プログラム言語における意味論と証明論

京大 数研 五十嵐滋

§1. 序

一つのプログラム言語における証明論とは，その言語で書かれるプログラムの持っている数学的性質を証明することに関する理論であって，Ianov，McCarthy，五十嵐などによって比較的早くから研究されていたが，近年アメリカを中心に急速な発展を見せはじめた。これに伴ってプログラム言語の意味論も，従来は「セマンティクスの記述」が多くの人々にとっての目標であったが，単なる「記述」から「証明論とかかわりを持つ記述」に注意が移りはじめて来たようである。

Aを§2以下で定められるALGOL型言語とするとき，この意味を表現する方法の典型的なものをいくつか挙げると次のようになる。（この分類の中にはMcCarthyの2階述語論理を用いた方法その他を入れてないので，不完全である。）

i) 各プログラムにactionの列を対応させる。

例. (イ) KnuthによるCFLの意味付け.

(ロ) Lucas等によるinterpreter的記述方法.

ii) 計算機の状態を抽象化して得られる空間を$\mathcal{D}$とし，$\mathcal{D}$から$\mathcal{D}$自身へのpartial mapping全体を$(\mathcal{D}\to\mathcal{D})$と書くことにする。$J:\mathcal{A}\longrightarrow(\mathcal{D}\to\mathcal{D})$なるJを具体的に定義することによって，$A\in\mathcal{A}$なるプログラムAの意味が$J(A)\in(\mathcal{D}\to\mathcal{D})$により表現されるとする。

例. (ハ) 五十嵐が公理的方法を提示する時用いたモデル.

(ニ) D. Scott等による記述.

(McCarthy等によるコンパイラの正当性の証明の背景にも，類似の面がある.)

iii) 上述のJに基いて，$\mathcal{A}$で定義される適当な2項関係α，(例えば:

$$\forall x(x\in\mathcal{D}\supset J(A)(x)=J(B)(x)) \qquad (1.1)$$

を$A\alpha B$とする)，を考え，それを表現する形式的体系によって意味が表現されたとする。

例. (ホ) 五十嵐，de Bakker等の公理的方法.

iv) $\mathcal{L}$を述語論理の意味での言語とするとき，適当な$\mathcal{L}$を固定し，写像$K:\mathcal{A}\longrightarrow\mathcal{L}$により$\mathcal{A}$の意味が定まるとする。通常$\mathcal{A}$で定義される(単項)述語$\beta$，(例えば:

$$\forall x\,(x\in\mathfrak{D}\supset J(A)(x)\ \underline{\text{is defined}}),\qquad (1.2)$$

を$\beta(A)$としたり,

$$\forall x\,(x\in\mathfrak{D}\supset p(J(A)(x)))\qquad (1.3)$$

を$\beta(A)$, 厳密には$\beta_p(A)$, として取上げる), を考えるとき

$K(A)$と$\beta(A)$が論理的に同値になる

ようにKが定められる。

例. (ハ) Engelerの$L_{\omega_1\omega}$による表現 ((1.2)に基く).

(ト) Floyd-Mannaによる1階述語論理での処理.

上述の諸方法の間には, 直観的には明らかな関係がいくつかあるが, それについて厳密に数学的考察をほどこしたものはまだ少い。(公理的方法においては, 当然モデルとの対比が考察されている。ほかにCooperやMannaの仕事もある。)

最初に述べたように, 意味論は証明論の影響を強く受けはじめたが, それでは証明論は意味論にとってかわることができるだろうか。(少くとも(iii)や(iv)は証明論そのものを意味論として位置づけようとする考え方である。) これについてはMcCarthyの所見に基いて,「証明論は, 一つの言語についてのコンパイラの正当性を証明し得るとき, その言語の意味論として位置づけ得る。」というあたりが穏当な所と

思われる。

同値関係に基く公理的方法によれば，実は(1.2)や(1.3)の関係も表現できることが分っている。(この意味では，Floydの与えた各コマンドの意味付けは必らずしも新しいものではない。) すなわち(iii)と(iv)の関係はかなり明確である。また(iii)と(ii)との関係は，(iii)の研究の根本的な部分にある。しかし(iii)と(i)との関係は，意味論における公理的方法が導入されたときから指摘されていたが，それは数学的な論証というほどの仕事をともなったものでなかった。それにこの問題は定式化できるという保証もない。が，一つの解答としては，(i)の方法に準じて定義されるactionの列が元のプログラムと同値であることを(iii)の方法を基礎にして証明して見せること，が考えられる。そして明らかにこれはコンパイラなりインタープリタなりの正当性を証明することと本質的に同じ問題である。

いうまでもなくコンパイラの正当性の証明は，プログラム言語における証明論の実用化への第一歩となるものであってそれ自身をテーマとする論文も近年いくつか見られるようになっている。以下，上の文中で下線をほどこした所の方針に従って具体的に論ずる。

§2. ALGOL型ステートメントの形成

Formation of Algol-like Statements

Alphabet

Let $\mathfrak{L}$, V, $\mathfrak{F}$, and $\mathfrak{R}$ be four disjoint sets whose elements are called label symbols, variable symbols, function symbols, and predicate symbols, respectively. The set $\mathfrak{F}$ is the union of disjoint sets $\mathfrak{F}^{(0)}, \mathfrak{F}^{(1)}, \ldots,$ and the elements of $\mathfrak{F}^{(n)}$ are called n-ary function symbols. Similarly, $\mathfrak{P}$ is the union of disjoint sets $\mathfrak{P}^{(0)}, \mathfrak{P}^{(1)}, \ldots,$ and the elements of $\mathfrak{P}^{(n)}$ are called n-ary predicate symbols. The alphabet of Algol-like statements consists of all the elements of $\mathfrak{L}$, V, $\mathfrak{F}$, and $\mathfrak{P}$, together with the following special symbols.

$$\Lambda \quad \text{-l} \quad := \quad ; \quad (\quad \rightarrow \quad , \quad)$$

In some cases described below the logical symbols:

$$\neg \quad \wedge \quad \vee \quad \forall \quad \exists$$

will be also contained.

Algol-like Statements

Algol-like statements, or statements, are defined together with a function denoted by $(\ )^{-}$ which sends each statement onto a finite subset of $\mathfrak{L}$, by generalized inductive definition as follows.

Atomic Statements

(a1) Λ is an atomic statement. $(\Lambda)^- = \emptyset$.

(a2) For each $\sigma \epsilon \mathfrak{L}$, σ and σ^{-1} are both atomic statements.
$(\sigma)^- = \emptyset$. $(\sigma^{-1})^- = \{\sigma\}$.

(a3) For each $x \epsilon V$ and each $y \epsilon V$, $x := y$ is an atomic statement.
$(x := y)^- = \emptyset$.

Statements

An atomic statement is a statement. Any other word on the above alphabet is a statement if and only if it is defined to be a statement by a repeated use of the following rules.

(b1) If A and B are two statements such that $(A)^- \cap (B)^- = \emptyset$, then A;B is a statement. $(A;B)^- = (A)^- + (B)^-$.

(b2) If $x := f_1, \ldots, x := f_n$ are n statements and $\pi^{(n)} \epsilon \mathfrak{P}^{(n)}$, then $x := \pi^{(n)} f_1 \ldots f_n$ is a statement. $(x := \pi^{(n)} f_1 \ldots f_n)^- = \emptyset$.

(b3) If $x := f_1, \ldots, x := f_n$, A , and B are n+2 statements such that $(A)^- \cap (B)^- = \emptyset$ and $\rho^{(n)} \epsilon \mathfrak{P}^{(n)}$, then $(\rho^{(n)} f_1 \ldots f_n \rightarrow A,B)$ is a statement. $((\rho^{(n)} f_1 \ldots f_n \rightarrow A,B))^- = (A)^- + (B)^-$.

A statement which is defined to be so only by the above rules will be called a basic statement.

(c1) If $(p \rightarrow A,B)$ is a statement, then $(\neg p \rightarrow A,B)$ is a statement.
$((\neg p \rightarrow A,B))^- = ((p \rightarrow A,B))^-$.

(c2) If $(p \to A,B)$ and $(q \to A,B)$ are two statements, then $(p \wedge q \to A,B)$ and $(p \vee q \to A,B)$ are both statements. The values of $(\ )^-$ are both identical with $((p \to A,B))^-$.

(c3) If $(p \to A,B)$ is a statement such that $x \in V$ occurs in p and neither $\forall x$ nor $\exists x$ occurs in p, then $(\forall x p \to A,B)$ and $(\exists x p \to A,B)$ are both statements. The values of $(\ )^-$ are both identical with $((p \to A,B))^-$.

Parentheses and commas will be used also auxiliarly to avoid syntactic ambiguity and to improve readability. Especially $\pi^{(n)} f_1 \ldots f_n$ and $\rho^{(n)} f_1 \ldots f_n$ are written as $\pi^{(n)}(f_1,\ldots,f_n)$ and $\rho^{(n)}(f_1,\ldots,f_n)$, respectively. Semicolons will be abbreviated if there is no possibility of ambiguity.

Representation by ALGOL 60

The statements in the above sense are intended to mean the statements in the sense of ALGOL 60 (Naur et al., 1960) as follows.

Λ corresponds to a dummy statement (empty).

σ corresponds to go to σ.

σ^{-1} corresponds to σ : (dummy statement labelled by σ).

$(p \to A,B)$ corresponds to if p then A else B.

:= , ; , $\neg$, $\wedge$, and $\vee$ mean the same as in ALGOL 60.

The parentheses used to avoid ambiugity either correspond to begin and end delimiting compound statements or mean the same as in ALGOL 60.

$(A)^-$ denotes the set of labels standing in A.

§3. プログラムのカテゴリー
Category of Programs

Programs in the General Sense

It seems to be convenient for us to consider more general programs as the background for the treatments of the properties of Algol-like statements. By a program, let us mean a partial function from an arbitrary set to another set together with its denotation. This definition does not exclude those partial functions which cannot be defined effectively. Instead, we shall describe it explicitly whenever the definability or constructiveness matters.

Programs will be denoted by $A,B,C,\ldots$. For each A , $J[A]$ denotes the partial function corresponding to A , and $G[A]$ the graph of $J[A]$. Let D be an Algol-like statement such that $D\in\mathcal{A}_U$, and $(U,\kappa,\mathfrak{R},\Gamma^o,J)$ be an interpretation. Then the pair $(D, (U,\kappa,\mathfrak{R},\Gamma^o,J))$ is a program, for a unique partial function $J[D]$, namely $D_{\mathfrak{B}}$, is determined by it. Therefore we shall assume the interpretation is fixed hereafter, so that each $D\in\mathcal{A}_U$ represents a unique program. Thus we identify an Algol-like statement with the program represented by it, and the set of such programs will be denoted by $\mathcal{Al}$.

What we shall do firstly is almost the same as considering a sub-category of $\mathcal{E}ns$ (the category of sets) whose objects are graphs of partial functions. The only difference lies in that the denotations are distinguished in our treatments. For instance, we do not say A and B are identical nor $A = B$, even if $J[A] = J[B]$, while we may say A and B are isomorphic.

Category $\mathcal{Pr}$

Each program will be called an object of category $\mathcal{Pr}$. The class of all the objects, namely programs, is denoted by $Ob\ \mathcal{Pr}$ For each pair A and B belonging to $Ob\ \mathcal{Pr}$, $Hom_{\mathcal{Pr}}(A,B)$ denotes the set of triples of the form (A,ζ,B) such that

$$\zeta : G[A] \to G[B]$$

and that ζ is a total function. The elements of $Hom_{\mathcal{Pr}}(A,B)$ are called morphisms of $\mathcal{Pr}$. If there is no possibility of confusion the morphism (A,ζ,B) will be abbreviated by ζ . We frequently write $\zeta : A \to B$ or $A \xrightarrow{\zeta} B$ instead of $\zeta \in Hom_{\mathcal{Pr}}(A,B)$. If $A \xrightarrow{\xi} B \xrightarrow{\eta} C$, then $(A,\eta\xi,C) \in Hom_{\mathcal{Pr}}(A,C)$ is defined as the composition of morphisms (A,ξ,B) and (B,η,C) , where $\eta\xi$ in $(A,\eta\xi,C)$ denotes the composition of functions ξ and η in the usual sense. Let $id_{G[A]}$ denote the identity function of $G[A]$ onto itself. The morphism $(A,id_{G[A]},A)$ is called the identity morphism of A and is denoted by 1_A .

We shall see that $\mathcal{Pr}$ satisfies the axioms of category as follows:

1. Associativity of Composition. If

$$A \xrightarrow{\xi} B \xrightarrow{\eta} C \xrightarrow{\zeta} D \ ,$$

then $\zeta(\eta\xi) = (\zeta\eta)\xi$ as morphisms.

2. Identity. If $A \xrightarrow{\xi} B$, then $\xi = \xi 1_A$. If $C \xrightarrow{\eta} A$, then $\eta = 1_A\eta$.

3. If the pairs (A_1,B_1) and (A_2,B_2) are distinct, then

$$\mathrm{Hom}_{\mathcal{P}r}(A_1,B_1) \cap \mathrm{Hom}_{\mathcal{P}r}(A_2,B_2) = \emptyset \ .$$

Category $\mathcal{P}r^{\iota}$

Let $\mathcal{P}r^{\iota}$ denote the full subcategory of $\mathcal{P}r$ such that $\mathrm{Ob}\,\mathcal{P}r^{\iota}$ consists of only those programs A such that

$$\mathrm{Dom}(J[A]) \subseteq |\mathcal{B}^{\iota}| \ ,$$

where

$$|\mathcal{B}^{\iota}| = \{a \,|\, a \in |\mathcal{B}| \text{ and } a_{\chi} = \iota\} \ . \quad \text{(See the below modification of J.)}$$

For each $A \in \mathrm{Ob}\,\mathcal{P}r^{\iota}$ and $B \in \mathrm{OB}\,\mathcal{P}r^{\iota}$,

$$\mathrm{Hom}_{\mathcal{P}r^{\iota}}(A,B) = \mathrm{Hom}_{\mathcal{P}r}(A,B) \ ,$$

by definition (of full subcategory).

We consider a map:

$$\mathrm{Ob}\,\mathcal{P}r \to \mathrm{Ob}\,\mathcal{P}r^{\iota}$$

which sends each $A \in \mathrm{Ob}\,\mathcal{P}r$ onto ${}_{\iota}A \in \mathrm{Ob}\,\mathcal{P}r^{\iota}$ such that

$$J[{}_{\iota}A] = J[A] \ |\mathcal{B}^{\iota}| \ .$$

That is to say we shall forget computational processes starting from any entry different from the normal one, namely the leftmost point, if A is an Algol-like program, modifying $J[A]$ into $J[A] \ |\mathcal{B}^{\iota}|$.

Hereafter we shall be concerned with $\mathcal{P}r^{\mathcal{L}}$, so that $A,B,C,\ldots$ will be understood as ${}_{\mathcal{L}}A, {}_{\mathcal{L}}B, {}_{\mathcal{L}}C,\ldots$ if the former do not belong to $\mathrm{Ob}\,\mathcal{P}r^{\mathcal{L}}$. Apparently the morphism (A,ζ,B) is a monomorphism, epimorphism, or isomorphism, according as the function ζ is univalent (1-1), onto, or univalent and onto. We shall write $\zeta : A \rightarrowtail B$ or $A \overset{\zeta}{\rightarrowtail} B$ to express that $\zeta : A \to B$ is an isomorphism, and $A \simeq B$ to express that there is an isomorphism from A to B, namely A and B are isomorphic.

Value-Preserving Monomorphisms

We pay special attention to such a monomorphism ζ that has the following property:

Suppose $\zeta : A \to B$, and the function $\zeta : G[A] \to G[B]$ sends $(a,b) \in G[A]$ onto $(c,d) \in G[B]$ such that

$$a = c$$

and

$$b_u = d_u \quad \text{for each } u \in X + \{X\},$$

for a subset X of U, for any $a \in \mathcal{B}^{\mathcal{L}}$

In such a case, ζ (as a morphism and as a function) will be said to preserve the values of X, or to preserve X, and we shall frequently write ζ_X instead of ζ in order to indicate that ζ preserves X. Moreover, if the choice of ζ itself does not matter, we write $A \vec{X} B$ instead of $\zeta_X : A \to B$. Similarly we shall frequently write $A \overset{\rightarrowtail}{X} B$ or $A \overset{\simeq}{X} B$ instead of $\zeta_X : A \rightarrowtail B$, and $A \cong B$ instead of $\zeta_U : A \rightarrowtail B$, that is $A \overset{\simeq}{U} B$.

§4. 公理系

Axioms and Inference Rules

Axiom 1.(a) $(AB)C \cong A(BC)$.

(b) $\sigma((AB)C) \cong \sigma(A(BC))$.

Axiom 2.(a) $A\Lambda \cong A$.

(b) $\Lambda A \cong A$. (Ie^{+})

Axiom 3.(a) $\sigma^{-1} \cong \Lambda$.

(b) $\sigma\sigma^{-1} \cong \Lambda$.

Axiom 4. $\sigma A \cong \sigma$.

$\sigma \notin A^{-}$.

Axiom 5. $A\Delta \cong \Delta$.

$A^{++} = \emptyset$.

Axiom 6. $x := x \cong \Lambda$. (Ia^{+})

Axiom 7.(a) $x := f;A;x := g \cong A_x[f]^{\circ};x := g_x[f]^{\circ}$. (Ib^{+})

$A^{++} = \emptyset$.

$L[A] \cap (V[f] \cup \{x\}) = \emptyset$.

(b) $x := f;A;y := g \cong x := f;A_x[f];y := g_x[f]$. (Ic^{+})

x and y are distinct.

$L[A] \cap (V[f] \cup \{x\}) = \emptyset$.

$x \notin V[f]$.

Axiom 8. $A \tilde{X} \Lambda$. (Id$^+$)

$L[A] \cap X = \emptyset$.

$A^{++} = \emptyset$.

$A^{-+} = \emptyset$.

Every function or predicate symbol occurring in A represents a total function or predicate, by the interpretation.

Axiom 9. $(1 \to A,B) \cong A$. (IIIm$^+$)

Axiom 10. $(p \to A,B) \cong (\neg p \to B,A)$. (IIIo$^+$)

Axiom 11.(a) $(p \to (q \to A,B),C) \cong (p \wedge q \to A,(p \wedge \neg q \to B,C))$. (IIIp$^+$)

(b) $(p \to (q \to A,B),C) \cong (p \to \Delta,C)$.

$p \supset \nabla q$.

Axiom 12.(a) $(p \to A,B)C \cong (p \to AC,BC')$. (IIIu$^+$)

(b) $\sigma(p \to A,B)C \cong \sigma(p \to AC,BC')$.

$\sigma \not\in C'^-$.

Axiom 13. $x := f;(p \to A,B) \cong (p_x[f]^* \to x := f;A,x := f;B)$. (IIIt$^+$)

* If $x \in V[f]$, then $p_x[f]$ is restricted to be $p_x[f]^\circ$.

Axiom 14. $(p \to A,B) \cong (p \to A_C[(p \to C,D)],B)$.

$L[A] \cap V[p] = \emptyset$.

Axiom 15.(a) $(p \to x := f, A) \cong (p \to x := g, A)$.

$p \supset f = g$.

(b) $(p \to x := f, A) \cong (p \to \Delta, A)$.

$p \supset \nabla f$.

Axiom 16.(a) $A \cong A_f[g]$.

$f = g$.

(b) $A \cong A_p[q]$.

$p \equiv q$.

Inference Rule 1.

$$\frac{A \underset{X}{\cong} B}{B \underset{X}{\cong} A} \quad . \tag{Ij$^+$}$$

Inference Rule 2.

$$\frac{A \underset{X}{\cong} B \quad B \underset{X}{\cong} C}{A \underset{X}{\cong} C} \quad . \tag{Ik$^+$}$$

Inference Rule 3.

$$\frac{A \underset{X}{\cong} B \quad A \underset{Y}{\cong} B}{A \underset{Z}{\cong} B} \quad .$$

$Z \subseteq X \cup Y$.

Inference Rule 4.

$$\frac{(p \to A,C) \underset{X}{\cong} (p \to B,C) \quad (q \to A,D) \underset{X}{\cong} (q \to B,D)}{(r \to A,E) \underset{X}{\cong} (r \to B,E)} \ .$$

$$r \supset p \vee q \ .$$

Inference Rule 5. (a)

$$\frac{\sigma A \cong \tau B}{C \cong C_{\sigma}[\tau]} \ .$$

A and B end with go-tos.

A and B occur in C .

(b)

$$\frac{\sigma A \cong B}{C \cong C_{\sigma}[B]} \ .$$

B ends with a go-to.

A occurs in C , or, A is $C_{A_1 \ldots A_n}[\Lambda,\ldots,\Lambda]$, where $A_1,\ldots,A_n$ are preceded by go-tos in C .

Inference Rule 6.

$$\frac{A \underset{X}{\cong} B}{AC \underset{XU\underline{L}[C]}{\cong} BC} \ .$$

$R[C] \subseteq X$.

$C^{++} \cap A^{-} = C^{++} \cap B^{-} = \emptyset$.

Inference Rule 7.

$$\frac{A \underset{X}{\cong} B^{*} \quad \sigma_i A \underset{X}{\cong} \sigma_i B}{CA \underset{X}{\cong} CB} \ .$$

$$C^{++} \cap A^{-} = C^{++} \cap B^{-} = \{\sigma_1, \ldots, \sigma_n\} \ .$$

$$A^{++} \cap C^{-} = B^{++} \cap C^{-} = \emptyset \ .$$

* If C ends with a go-to, or A and B both begin with labellings, then the upper left formula may be omitted, provided that $n \geq 1$.

Inference Rule 8.

$$\frac{A \cong B^{*} \quad \sigma_i A \cong \sigma_i B}{C \cong C_A[B]} \ . \qquad (\text{IVg}^{+})$$

$$C_A[\Lambda]^{++} \cap A^{-} = C_A[\Lambda]^{++} \cap B^{-} = \{\sigma_1, \ldots, \sigma_n\} \ .$$

* Same as above.

Inference Rule 9.

$$\frac{D^i A \cong A^i A \quad \tilde{D}^i B \cong B^i B \quad A^i_{\sigma_1 \ldots \sigma_n}[\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n]^{\circ} \underset{X}{\cong} B^i \quad B^i \underset{X}{\cong} C^i}{D^k A \underset{X}{\cong} \tilde{D}^k B} \ .$$

$$k \epsilon [n] \ .$$

1. The set $S = \{\sigma_1,\ldots,\sigma_n\}$ is a non-empty subset of A^-, and a total function

$$\zeta : S \to \mathfrak{L}$$

sends each σ_i onto $\tilde{\sigma}_i$. ζ, together with S, satisfies the following conditions:

$$S \supseteq S'$$

and

$$\zeta(\sigma) = \sigma \quad \text{for each } \sigma \in S \cap S'',$$

where

$$S' = \bigcup_i (A^i)^{++} \cap A^-$$

and

$$S'' = \bigcup_i (A^i)^{++} \cap A^{++} .$$

2. The following conditions are satisfied for each $i \in [n]$.

(i) D^i is of the form $(p_i \to \sigma_i, \delta^i)$ and $\tilde{D}^i$ is of the form $(p_i \to \tilde{\sigma}_i, \delta^i)$, where δ^i is either τ_i or $\tau_i^{-1}\tau_i$ such that $\tau_i \notin A^{\pm} \cup B^{\pm}$.

(ii) All the occurrences of σ_i in $A^1,\ldots,A^n$ are within the statements of the form $(p_i \to \sigma_i, \varepsilon^i)$, or all the occurrences of $\tilde{\sigma}_i$ in $B^1,\ldots,B^n$ are within the statements of the form $(p_i \to \tilde{\sigma}_i, \varepsilon^i)$ where ε^i subjects to the same restriction as δ^i above.

(iii) $R[C^i] \subseteq X$.

3. If A does not begin with a labelling σ^{-1} such that $\sigma \in S$, then all of $A^1,\ldots,A^n$ must end with go-tos. If B does not begin with a labelling σ^{-1} such that $\sigma \in \zeta(S)$, then all of $B^1,\ldots,B^n$ must end with go-tos.

§5. プログラムの分解
Decomposition of Statements

Let V be a subset of $\mathcal{V}$ such that $\mathcal{V}-V$ contains infinite elements $w_0, w_1, \ldots$, and L be a subset of $\mathfrak{L}$ such that $\mathfrak{L}-L$ contains infinite elements $\sigma_0, \sigma_1, \ldots$. By $\mathcal{A}_0$ is denoted the set of statements defined by induction as follows.

(d1) Λ belongs to $\mathcal{A}_0$.

(d2) For each $\sigma \in \mathfrak{L}$, σ and σ^{-1} belong to $\mathcal{A}_0$.

(d3) For each $x \in \mathcal{V}$ and a fixed element w_0 of $\mathcal{V}-V$, $x := w_0$ and $w_0 := x$ belong to $\mathcal{A}_0$.

(d4) For each $\pi^{(n)} \in \mathfrak{F}^{(n)}$ and $e_1, \ldots, e_{n-1}$ such that either $e_i \in \mathfrak{F}^{(0)}$ or $e_i \in \mathcal{V}$ for each $i \in [n-1]$, $w_0 := \pi^{(n)} w_0 e_1 \cdots e_{n-1}$ belongs to $\mathcal{A}_0$.

(d5) For each $\rho^{(n)} \in \mathfrak{P}^{(n)}$, $\sigma \in \mathfrak{L}$, and $e_1, \ldots, e_{n-1}$ as above, $(\rho^{(n)} w_0 e_1 \cdots e_{n-1} \to \sigma, \Lambda)$ belongs to $\mathcal{A}_0$.

(e1) If A and B belong to $\mathcal{A}_0$, then AB belongs to $\mathcal{A}_0$. ($A^- \cap B^- = \phi$ should be satisfied. Otherwise, AB is not a statement.)

Let $\mathcal{A}_1$ be the set of statements consisting of all A such that $V[A] \subseteq V$, $A^{\pm} \subseteq L$, and that the logical symbols other than $\neg$ and $\vee$ do not occur in A.

We shall establish a function

$$\Phi : \mathcal{A}_1 \to \mathcal{A}_0 ,$$

which has the following characteristics.

1. Constructiveness:

Φ is total and effectively defined.

2. Correctness:

$\vdash A \overset{\approx}{V} \Phi(A)$ for any $A \in \mathcal{A}_1$.

In other words, Φ is an algorithm that carries out a translation of $\mathcal{A}_1$ into $\mathcal{A}_0$, of which the latter consists of sequences of relatively simple statements. Moreover, we can formally prove that Φ always gives a statement equivalent to the original one in so far as the values of variables belonging to V and the destinations of exits are concerned. (Actually we prove the above also for each entry. cf. proof of Theorem 26).

For the convenience of description, we introduce two sets of statements, as follows:

$$\mathcal{A}_2 = \{x := f \mid x \in V \text{ and } V[f] \subseteq V\} .$$

$$\mathcal{A}_3 = \{(p \to \tau, \Lambda) \mid \tau \in \mathfrak{L} \text{ and } V[p] \subseteq V\} .$$

Besides, $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ will be used, whose elements differ from $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$, respectively, only in that some suffixes are added. (See Definition of Θ below.)

Definition of Φ

Let Θ and Ψ be two functions as defined below. Then

$$\Phi(A) = \Psi(\Theta_0(A)) \quad \text{for each } A \in \mathcal{A}_1 .$$

1. Definition of Θ

We define the function

$$\Theta : \mathcal{A}_1 \times \eta \to \mathcal{A}_1^* ,$$

where the elements of $\mathcal{A}_1^*$ are statements whose symbols are possibly suffixed. For each A and each $\nu \in \eta$, $\Theta_\nu(A)$ denotes the image of (A,ν). Actually, however, Θ is extended so that, for each arithmetic expression f such that $V[f] \subseteq V$ and for each Boolean expression p such that $V[p] \subseteq V$, $\Theta_\nu(f)$ and $\Theta_\nu(p)$ are defined. Besides, two auxiliary functions

$$\lambda : \mathcal{A}_1 \cup \{p | V[p] \subseteq V\} \to \eta$$

and

$$\mu : \{f | V[f] \subseteq V\} \to \eta$$

are defined.

Practical meaning of these functions are as follows.

$\mu(f)$: The number of required working storages to compute f.

$\Theta_\nu(f)$: The result of suffixing function symbols occurring in f so as to specify the allocation of working storages. (ν is irrelevant.)

$\lambda(p)$: The number of auxiliary labels to compute p, which is the number of occurrences of symbol $\neg$ in p.

$\mu(p)$: The number of required working storages to compute p.

$\Theta_\nu(p)$: The result of suffixing p to specify all the auxiliary labels using index greater than ν .

$\lambda(A)$ and $\Theta_\nu(A)$: Similar to $\lambda(p)$ and $\Theta_\nu(p)$.

Functions Θ , λ , and μ are defined simultaneously by induction on statements as follows.

Atomic Statements

(a1) and (a2) $C = \Lambda$, σ, or σ^{-1} :

$\Theta_\nu(C) = C$ for each ν .

$\lambda(C) = 0$.

(a3) $C = x := f$, where $f = y$:

$\mu(f) = 0$.

$\Theta_\nu(f) = f$ for each ν .

$\Theta_\nu(C) = x := \Theta_\nu(f)$. (1)

$\lambda(C) = 0$. (2)

Statements (non-atomic)

(b1) $C = AB$:

$\Theta_\nu(C) = \Theta_\nu(A)\Theta_{\nu+\lambda(A)}(B)$

$\lambda(C) = \lambda(A) + \lambda(B)$.

(b2) $C = x := f$, where $f = \pi^{(n)} f_0 \ldots f_{n-1}$:

$$\mu(f) = M+m \ ,$$

where

$$M = \max_{0 \leq i \leq n-1} \mu(f_i) \ , \tag{3}$$

and m is the number of f_i such that $f_i \notin V$.

$$\Theta_\nu(f) = \pi^{(n)}_{M+1,\ldots,M+m} \Theta_\nu(f_0) \ldots \Theta_\nu(f_{n-1}) \ . \tag{4}$$

$\Theta_\nu(C)$ and $\lambda(C)$ are defined by (1) and (2) above.

(b3) $C = (p \to A,B)$, where $p = \rho^{(n)} f_0 \ldots f_{n-1}$:

$$\mu(p) = M+m \ ,$$

where M and m are defined by (3) and (4) above.

$$\Theta_\nu(p) = \rho^{(n)}_{M+1,\ldots,M+m} \Theta_\nu(f_0) \ldots \Theta_\nu(f_{n-1}) \ .$$

$$\lambda(p) = 0 \ .$$

(i) If A is τ and B is Λ , then

$$\Theta_\nu(C) = (\Theta_\nu(p) \to \tau,\Lambda) \ , \tag{5}$$

and

$$\lambda(C) = \lambda(p) \ . \tag{6}$$

(ii) If A is not of the form τ or B is not Λ , then

$$\Theta_\nu(C) = (\Theta_{\nu+\lambda(A)+\lambda(B)}(p) \to_{N+1,N+2} \Theta_\nu(A), \Theta_{\nu+\lambda(A)}(B)) \ , \tag{7}$$

where

$$N = \nu + \lambda(A) + \lambda(B) + \lambda(p) \quad ,$$

and

$$\lambda(C) = N+2 \quad . \qquad (8)$$

(c1) $C = (\neg p \to A,B)$:

$$\Theta_\nu(\neg p) = \neg_{\nu+1} \Theta_\nu(p) \quad .$$

$$\lambda(\neg p) = \lambda(p)+1 \quad .$$

$\Theta_\nu(C)$ and $\lambda(C)$ are defined by (5)-(8) above. (Substitute $\neg p$ in place of p .)

(c2) $C = (p \vee q \to A,B)$:

$$\Theta_\nu(p \vee q) = \Theta_\nu(p) \vee \Theta_{\nu+\lambda(p)}(q) \quad .$$

$$\lambda(p \vee q) = \lambda(p) + \lambda(q) \quad .$$

$\Theta_\nu(C)$ and $\lambda(C)$ are defined by (5)-(8) above. (Subsitute $p \vee q$ in place of p .)

2. Definition of Ψ

We define the function

$$\Psi : a_1^* \cup a_2^* \cup a_3^* \to a_0 \quad .$$

By A^* , f^* , and p^* will be denoted $\Theta_\nu(A)$, $\Theta_\nu(f)$, and $\Theta_\nu(p)$, respectively, for certain values of ν . Thus, for instance,

(b1) below, i.e.,

$$\Psi(A^*B^*) = \Psi(A^*)\Psi(B^*)$$

reads as follows:

Since $C = AB$, $\Theta_\nu(C)$ is of the form A^*B^*. Define $\Psi(A^*)\Psi(B^*)$ as $\Psi(\Theta_\nu(C))$.

w_0 plays the role of an accumulator.

Ψ is defined by induction as follows.

Atomic Statements

(a1) $\Psi(\Lambda) = \Lambda$.

(a2) $\Psi(\sigma) = \sigma$.

$\Psi(\sigma^{-1}) = \sigma^{-1}$.

(a3) (i) $\Psi(w_0 := y) = w_0 := y$.

(ii) If $x \neq w_0$, then $\Psi(x := y)$ is defined by (1) below. (Substitute y in place of f.)

Statements (non-atomic)

(b1) $\Psi(A^*B^*) = \Psi(A^*)\Psi(B^*)$.

(b2) (i) $\Psi(w_0 := \pi^{(0)}) = w_0 := \pi^{(0)}$.

(ii) $\Psi(w_0 := \pi^{(n)}_{\alpha(1)\ldots\alpha(m)} f_0^* \ldots f_{n-1}^*)$

$= C_{n-1}\ldots C_0;\ w_0 := \pi^{(n)} w_0 u_1 \ldots u_{n-1}$, $(n \geq 1)$

where

$$u_i = \begin{cases} f_i & f_i \in V \\ w_{\alpha(\beta(i))} & f_i \notin V \end{cases} \qquad \text{for } i \in [n-1] \ ,$$

C_0 is $w_0 := f_0$,

and

$$C_i = \begin{cases} \Lambda & f_i \in V \\ u_i := f_i & f_i \notin V \end{cases} \qquad \text{for each } i \in [n-1] \ ,$$

$\beta(i)$ being defined by the following induction:

$$\beta(0) = 0 \ .$$

$$\beta(i+1) = \begin{cases} \beta(i) & f_i \in V \\ \beta(i)+1 & f_i \notin V \end{cases} \ .$$

(iii) $\Psi(x := f^*) = \Psi(w_0 := f^*)x := w_0 \ . \quad (x \neq w_0)$ (1)

(b3) (i) $\Psi((\rho^{(0)} \to \tau, \Lambda)) = (\rho^{(0)} \to \tau, \Lambda) \ .$

(ii) $\Psi((\rho^{(n)}_{\alpha(1)\ldots\alpha(m)} f_0^* \ldots f_{n-1}^* \to_{\gamma(1)\gamma(2)} \tau, \Lambda))$, $(n \geq 1)$

$= C_{n-1} \cdots C_0 (\rho^{(n)} w_0 u_1 \cdots u_{n-1} \to \tau, \Lambda)$

where $C_0, \ldots, C_{n-1}$, $u_1, \ldots, u_{n-1}$ are the same as above.

(cf. (b2)(ii).)

(iii) $\Psi((p^* \to_{\gamma(1)\gamma(2)} A^*, B^*)$

$= \Psi((p^* \to_{\gamma(1)\gamma(2)} \sigma_{\gamma(1)}, \Lambda))\Psi(B^*)\sigma_{\gamma(2)}\sigma^{-1}_{\gamma(1)}\Psi(A^*)\sigma^{-1}_{\gamma(2)} \ .$ (2)

(A is not of the form τ , or B is not Λ .)

(c1) (i) $\Psi((\neg_\delta p^* \to \tau, \Lambda))$

$$= \Psi((p^* \to \sigma_\delta, \Lambda))\tau\sigma_\delta^{-1} \quad .$$

(ii) If A is not of the form τ, or B is not Λ, then $\Psi((\neg_\delta p^* \to_{\gamma(1)\gamma(2)} A,B))$ is defined by (2) above. (Substitute $\neg_\delta p^*$ in place of p^*.)

(c2) (i) $\Psi((p^* \vee q^* \to \tau, \Lambda))$

$$= \Psi((p^* \to \tau, \Lambda))\Psi((q^* \to \tau, \Lambda)) \quad .$$

(ii) If A is not of the form τ, or B is not Λ, then $\Psi((p^* \vee q^* \to_{\gamma(1)\gamma(2)} A,B))$ is defined by (2) above. (Substitute $p^* \vee q^*$ in place of p^*.)

Example

We consider the statement

$$\underline{\text{if}}\ x < 0\ \underline{\text{then}}\ x := -x\ , \tag{1}$$

which was used as an example of compilation in (Igarashi, 1968).

Here, let us allow only binary $-$, and see how the statement

$$\underline{\text{if}}\ x < 0\ \underline{\text{then}}\ x := 0-x\ , \tag{2}$$

namely

$$(x < 0 \to x := 0-x, \Lambda) \tag{3}$$

in our notation, is treated.

Let A be $(\rho^{(1)}x \to x := \pi^{(2)}\pi^{(0)}x, \Lambda)$. Then,

$$\Theta_0(A) = A^* = (\rho^{(1)}x \rightarrow_{1,2} x := \pi_1^{(2)}\ \pi^{(0)}x,\Lambda)$$

and

$$\Psi(A^*) = w_0 := x;(\rho^{(1)}w_0 \rightarrow \sigma_1,\Lambda)\sigma_2\sigma_1^{-1}\ ;$$

$$w_0 := \pi^{(0)};\ w_0 := \pi^{(2)}w_0x;\ x := w_0;\ \sigma_2^{-1}\ .$$

Especially, we define $x < 0$ as $\rho^{(1)}x$, 0 as $\pi^{(0)}$, and $x-y$ as $\pi^{(0)}xy$, so that A becomes (3).

For readability's sake, $\Phi(A)$ i.e., $\Psi(A^*)$ will be written in ALGOL 60 and listed with corresponding actions, symbols w_0, σ_1, and σ_2 being replaced by acc, L1, and L2, respectively.

```
    acc := x;                          load x
    if acc < 0 then go to L1;          jump on minus L1
    go to L2;                          jump L2
L1:                                    insert label L1

    acc := 0;                          load 0
    acc := acc - x;                    subtract x
    x := acc;                          store x
L2:                                    insert label L2
```

(4)

Statement (4) is different only in trivial points from program *B* (in the above paper) for which

$$\vdash (1) \underset{\{x\}}{\simeq} \beta$$

is proved as an example of derivation. That proof, for this particular pair of statements, needed two pages of derivation (20 steps) preceded by one page (10 steps) for an auxiliary formula, being derived directly from the previous formal system. In the present paper, however, we shall prove, also formally, that

$$A \underset{V}{\simeq} \Phi(A)$$

is valid for every $A \in \mathcal{A}_1$, which implies that $(2) \underset{V-\{acc\}}{\simeq} (4)$.

参考文献

・本稿は下記の論文に基いている。

S. Igarashi, Semantics of Algol-like Statements, Semantics of Algorithmic Languages, (E. Engeler, ed.), Lecture Note in Mathematics, vol. 188, Springer Verlag Berlin-heidelberg-New York, (1971), pp. 117-177. Also, Stanford Artificial Intelligence Project Memo AIM-129, or Computer Science Department Report No. CS167, Stanford University, (1970).

それ以外の文献は次ページ以下のとおり。

References

Cooper, D. C. (1969), Program Scheme Equivalence and Second-Order Logic, Machine Intelligence 4, Edinburgh University Press, pp. 3-15.

de Bakker, J. W. (1968), Axiomatics of Simple Assignment Statements, Report MR94, Mathematisch Centrum, Amsterdam.

_____ (1969), Semantics of Programming Languages, Advances in Information Systems Science 2, Plenum Press

Engeler, E. (1967), Algorithmic Properties of Structures, Math. Systems Theory 1, pp. 183-195.

Floyd, R. W. (1967), Assigning Meanings to Programs, Proc. of Symposia in Applied Mathematics 19, pp. 19-32.

Hoare, C. A. R. (1969), An Axiomatic Basis for Computer Programming, Communication of the Assoc. for Computing Machinery 12, No. 10, pp. 576-583.

Igarashi, S. (1963), On the Logical Schemes of Algorithms, Information Processing in Japan 3, pp. 12-18.

_____ (1964), An Axiomatic Approach to the Equivalence Problems of Algorithms with Applications, Ph.D. Thesis, University of Tokyo; also
Report of the Computer Centre, University of Tokyo 1 (1968), pp. 1-101; and
Publications of the Research Institute for Mathematical Sciences, Kyoto University B, No. 34 (1969).

_____ (1968), On the Equivalence of Programs Represented by Algol-like Statements, Report of the Computer Centre, University of Tokyo 1, pp. 103-118; also Publications of the Research Institute for Mathematical Sciences, Kyoto University B, No. 33 (1969).

Kaplan, D. M. (1968), The Formal Theoretic Analysis of Strong Equivalnece for Elemental Programs, Ph.D. Thesis, Stanford University.

Knuth, D. E. (1968), Semantics of Context-Free Languages, Math. Systems Theory 2, pp. 127-145.

Lucas, P. et al. (1968), Method and Notation for the Formal Definition of Programming Languages, Technical Report TR 25.087, IBM Laboratory, Vienna.

Lukasiemicz, J. (1941), Die Logik und das Grundlagenproblem, Les Entretiens de Zurich sur les Fondements et la Methode des Sciences Mathematiques, pp. 82-108. (Or, see the below article.)

_____ (1929, English translation of the second edition: 1963), Elements of Mathematical Logic, Pergamon Press, Oxford.

Manna, Z. (1968), Termination of Algorithms, Ph.D. Thesis, Carnegie-Mellon University.

_____ (1969), The Correctness of Programs, Journal of Computer and System Sciences 3, No. 2, pp. 119-127.

_____ and McCarthy, J. (1969), Properties of Programs and Partial Function Logic, Stanford Artificial Intelligence Project Memo AIM-100, Stanford Univ. Also in Machine Intelligence 5 (1970), Edinburgh U.Press.

McCarthy, J. (1963a), A Basis for a Mathematical Theory of Computation, Computer Programming and Formal Systems, North-Holland Publishing Co., Amsterdam, pp. 33-69.

_____ (1963b), Predicate Calculus with "Undefined" as a Truth-Value, Stanford Artificial Intelligence Project, Memo 1.

_____ and Painter, J. (1967), Correctness of a Compiler for Arithmetic Expressions, Proc. of Symposia in Applied Mathematics 19, pp. 34-41.

Naur, P. et al. (1960), Report on the Algorithmic Language ALGOL 60, Communication of Assoc. for Computing Machinery 3, pp. 299-314.
Also, Revised Report on the Algorithmic Language ALGOL 60, Communication of the Assoc. for Computing Machinery 6 (1963), pp. 1-17.

Painter, J. A. (1967), Semantic Correctness of a Compiler for an ALGOL-like Language, Ph.D. Thesis, Stanford University

Rutledge, J. D. (1964), On Ianov's Program Schemata, Journal of Assoc. Computing Machinery 11, pp. 1-9.

Strachey, C. and Scott, D. (1970), Mathematical Semantics for Two Simple Languages, IFIP WG2.2, Boston.

Yanov, Y. I. (1958, English edition: 1960), The Logical Schemes of Algorithms, Problems of Cybernetics 1, Pergamon Press, New York, pp. 82-140.